UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/901,806	07/10/2001	Keith R. Slavin	303.743US1	7913

21186 7590 07/14/2005

SCHWEGMAN, LUNDBERG, WOESSNER & KLUTH, P.A.
P.O. BOX 2938
MINNEAPOLIS, MN 55402-0938

EXAMINER	
RAMPURIA, SATISH	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 07/14/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/901,806	SLAVIN ET AL.	
	Examiner	Art Unit	
	Satish S. Rampuria	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *16 May 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-51* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-51* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date 05/16/05
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

Response to Amendment

1. This action is in response to the amendment received on 05/16/2005.

2. The objection to claim 22 due to word (an) appeared twice in the claim is withdrawn in view of applicant's amendment.

3. The rejections under 35 U.S.C. §112 first paragraph to claim 1, 6, 14, and 42 is withdrawn in view of applicant's amendment.

4. Claims amended by the applicant: 1, 7, 14, 22, 34, 38, 42 and 48.

5. Claims pending in the application: 1-51.

Information Disclosure Statement

6. An initialed and dated copy of Applicant's e-IDS form 1449 filed on 05/16/2005 is attached to the instant Office action.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1-51 are rejected under 35 U.S.C. 103(a) as being unpatentable over US Patent No. 6,014,723 to Tremblay et al. (hereinafter called Tremblay) in view admitted prior art and further in view of US Patent No. 6,240,499 to Spencer.

Per claim 1:

Tremblay disclose:

- A method of accessing a memory array (col. 3, line, 21-22 "a fully associative memory, such as a content addressable memory") implemented in a computer-readable medium comprising:
- providing data contained within a one-dimensional array of allocated memory (col. 3, lines 49-51 "memory locations is configured to store one pair of the array size values and is associated with ... memory locations");
- dynamically declaring a dimensional dynamic overlay on the data contained within the one-dimensional array (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions") from within a block of statements in a user-defined software program subroutine to initialize attributes within an array attribute storage object (col. 26, lines 58-59 "executing a sequence of translated instructions that define an array access boundary exception subroutine").

Tremblay does not explicitly disclose accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object.

However, admitted prior art discloses in an analogous computer system accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object (Applicant's specification, page 3, lines 14-16 "The array elements are

often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of accessing the data stored in an array using index as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by Tremblay. The modification would be obvious because of one of ordinary skill in the art would be motivated to use indexes to access data in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose wherein the dynamic overlay is defined within the subroutine and provides a dimensional view of the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose wherein the dynamic overlay is defined within the subroutine and provides a dimensional view of the one-dimensional array (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory"), the dimensional dynamic overlay being capable of

providing a view of at least two dimensions on the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of dynamic overlay is defined within the subroutine and provides a dimensional view of the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 2:

The rejection of claim 1 is incorporated, and further, Tremblay disclose:

- providing a pointer to the one-dimensional array of allocated memory (col. 23, lines 61-63 "the field block pointer associated with the stored index that matched in input index is output from the second section of the associative memory");
- providing an array access identifier (col. 3, line 60 "an array access instruction identifier"); and
- providing array information for the declared dimensional dynamic overlay (col. 3, lines 28-31 "to verify that each access of an information array is within a maximum array size boundary value and a minimum array size boundary value").

Per claim 3:

The rejection of claim 2 is incorporated, and further, Tremblay disclose:

- wherein providing array information includes providing array height information, array width information and array stride information. The limitation recited in this claim are similar to those recited in claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Per claim 4:

The rejection of claim 2 is incorporated, and further, Tremblay disclose:

- performing allocated memory offset and array index boundary calculations on the array information that is for the dimensional dynamic overlay. The limitation recited in this claim are similar to those recited in claim 2 and rejected under the same rational set forth in connection with the rejection of claim 2 above.

Per claim 5:

The rejection of claim 1 is incorporated, and further, Tremblay disclose:

- wherein declaring a dimensional dynamic overlay on the data contained within the one-dimensional array includes coding a dimensional dynamic overlay declaration using extended programming language from within the subroutine (col. 24, lines 55-59 "translates program information 609 into translated instructions that include array access

instructions on bus 611... each of which corresponds to one of the translated instructions. Each array access instruction references an element within the array").

Per claim 6:

The rejection of claim 1 is incorporated, and further, Tremblay disclose:

- setting an explicit boundary policy for the declared dimensional dynamic overlay (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value").

Per claim 7:

Tremblay disclose:

- A method of accessing a memory array (col. 3, line, 21-22 "a fully associative memory, such as a content addressable memory") implemented in a computer-readable medium, comprising:
- providing data contained within a one-dimensional array of allocated memory (col. 3, lines 49-51 "memory locations is configured to store one pair of the array size values and is associated with ... memory locations");
- dynamically declaring a dimensional dynamic overlay on the data contained within the one-dimensional array (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions") from within a block of statements in a user-

defined software program subroutine (col. 26, lines 58-59 "executing a sequence of translated instructions that define an array access boundary exception subroutine").

Tremblay does not explicitly disclose providing a dynamic overlay storage object associated with the declared dimensional overlay; assigning attributes from the declared dimensional dynamic overlay to the storage object; and accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object.

However, admitted prior art discloses in an analogous computer system providing a dynamic overlay storage object associated with the declared dimensional overlay (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared storage that is only available after its declaration, and only within the scope of a block of statements in which it is declared"); assigning attributes from the declared dimensional dynamic overlay to the storage object (Applicant's specification, page 4, lines 26-27 "Automatic storage is declared storage that is only available after its declaration"); and accessing the data from within the block of statements as a dimensional indexed array using the array attribute storage object (Applicant's specification, page 3, lines 14-16 "The array elements are often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of providing storage object, assigning attributes to the storage object, and accessing the data stored in an array using index as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by

Tremblay. The modification would be obvious because of one of ordinary skill in the art would be motivated to provide storage object, assign attributes to the storage object, and use indexes to access data in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose wherein the dynamic overlay is defined within the subroutine and provides a dimensional view on the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose wherein the dynamic overlay is defined within the subroutine and provides a dimensional view on the one-dimensional array (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory"), the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of dynamic overlay is defined within the

subroutine and provides a dimensional view on the one-dimensional array, the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 8:

The rejection of claim 7 is incorporated, and further, Tremblay does not explicitly disclose wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a hardware environment art.

However, admitted prior art discloses in an analogous computer providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a hardware environment art (Applicant's specification, page 6, lines 21-22 "In hardware, memory is accessed as one-dimensional contiguous storage that is indexed by the memory address").

The feature of having a hardware environment would be obvious for the reasons set forth in the rejection of claim 7.

Per claim 9:

The rejection of claim 7 is incorporated, and further, Tremblay does not explicitly disclose wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a software environment.

However, admitted prior art discloses in an analogous computer providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a software environment (Applicant's specification, page 6, lines 22-24 "In C, a base address is provided for an allocated region of memory, and then one-dimensional array access can be achieved by adding an index offset (scaled by the array element size) to the base address").

The feature of having a software environment would be obvious for the reasons set forth in the rejection of claim 7.

Per claim 10:

The rejection of claim 7 is incorporated, and further, Tremblay does not explicitly disclose automatically freeing the dynamic overlay storage object when leaving the block of statements in which the dimensional dynamic overlay was declared.

However, admitted prior art discloses in an analogous computer automatically freeing the dynamic overlay storage object when leaving the block of statements in which the dimensional dynamic overlay was declared (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared... storage is released when code execution leaves the block").

The feature of automatically freeing the dynamic overlay storage object would be obvious for the reasons set forth in the rejection of claim 7.

Per claim 11:

The rejection of claim 7 is incorporated, and further, Tremblay disclose:

- providing a reference to the one-dimensional array of allocated memory (col. 23, lines 61-63 "the field block pointer associated with the stored index that matched in input index is output from the second section of the associative memory");
- providing an array access identifier (col. 3, line 60 "an array access instruction identifier"); and
- providing array information for the declared dimensional dynamic overlay (col. 3, lines 28-31 "to verify that each access of an information array is within a maximum array size boundary value and a minimum array size boundary value").

Per claim 12:

The rejection of claim 11 is incorporated, and further, Tremblay disclose:

- wherein providing array information includes providing array height information, array width information and array stride information. The limitation recited in this claim are similar to those recited in claim 11 and rejected under the same rational set forth in connection with the rejection of claim 11 above.

Per claim 13:

The rejection of claim 11 is incorporated, and further, Tremblay disclose:

- performing allocated memory offset and array index boundary calculations on the array information for the dimensional dynamic overlay. The limitation recited in this claim are

similar to those recited in claim 11 and rejected under the same rational set forth in connection with the rejection of claim 11 above.

Per claim 14:

Tremblay disclose:

- A method of creating and accessing a multi-dimensional dynamic array (col. 3, line, 21-22 "a fully associative memory, such as a content addressable memory") implemented in a computer-readable medium, comprising:
- dynamically declaring a dimensional dynamic array from within a block of statements in a user-defined software program subroutine (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions");
- dynamically allocating memory storage sufficient to store all the elements for the declared dimensional dynamic array (col. 25, lines 19-23 "array access processor 612 is simply instructions that are executed by execution unit 616 to load the identifier, and minimum and maximum array size values into associative memory element 614").

Tremblay does not explicitly disclose providing a dynamic overlay storage object with attributes initialized from the dimensional dynamic array declaration; accessing data from the dynamically allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object; automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared; and

automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared.

However, admitted prior art discloses in an analogous computer system providing a dynamic overlay storage object with attributes initialized from the dimensional dynamic array declaration (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared storage that is only available after its declaration, and only within the scope of a block of statements in which it is declared"); accessing data from the dynamically allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object (Applicant's specification, page 3, lines 14-16 "The array elements are often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array"); automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared... storage is released when code execution leaves the block"); and automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared (Applicant's specification, page 4, lines 30-31 "Subroutines are used in C to dynamically allocate a contiguous region of memory and free it after it is no longer needed").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of providing a dynamic overlay storage object accessing data from the dynamically allocated memory, automatically freeing the dynamically allocated memory as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by Tremblay. The modification would be obvious because of one of

ordinary skill in the art would be motivated to providing a dynamic overlay storage object accessing data from the dynamically allocated memory, automatically freeing the dynamically allocated memory in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose wherein the dynamic overlay storage object provides a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay storage object being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose wherein the dynamic overlay storage object provides a dimensional view on a one-dimensional array associated with contiguous memory locations (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory") the dimensional dynamic overlay storage object being capable of providing a view of at least two dimensions on the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of dynamic overlay storage object provides a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay storage object being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 15:

The rejection of claim 14 is incorporated, and further, Tremblay disclose:

- providing a pointer to a corresponding one-dimensional array in the dynamically allocated memory storage (col. 23, lines 61-63 "the field block pointer associated with the stored index that matched in input index is output from the second section of the associative memory");
- providing a handle for an array access (col. 3, line 60 "an array access instruction identifier"); and
- providing array information (col. 3, lines 28-31 "to verify that each access of an information array is within a maximum array size boundary value and a minimum array size boundary value").

Per claim 16:

The rejection of claim 15 is incorporated, and further, Tremblay disclose:

- performing allocated memory offset and array index boundary calculations on the array information provided for the declared dimensional dynamic array. The limitation recited in this claim are similar to those recited in claim 15 and rejected under the same rational set forth in connection with the rejection of claim 15 above.

Per claim 17:

The rejection of claim 15 is incorporated, and further, Tremblay disclose:

- wherein providing array information includes providing array height information, array width information and array stride information. The limitation recited in this claim are similar to those recited in claim 15 and rejected under the same rational set forth in connection with the rejection of claim 15 above.

Per claim 18:

The rejection of claim 14 is incorporated, and further, Tremblay does not explicitly disclose wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a hardware environment art.

However, admitted prior art discloses in an analogous computer providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a hardware

environment art (Applicant's specification, page 6, lines 21-22 "In hardware, memory is accessed as one-dimensional contiguous storage that is indexed by the memory address").

The feature of having a hardware environment would be obvious for the reasons set forth in the rejection of claim 14.

Per claim 19:

The rejection of claim 14 is incorporated, and further, Tremblay does not explicitly disclose wherein providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a software environment.

However, admitted prior art discloses in an analogous computer providing a dynamic overlay storage object includes providing a dynamic overlay storage object within a software environment (Applicant's specification, page 6, lines 22-24 "In C, a base address is provided for an allocated region of memory, and then one-dimensional array access can be achieved by adding an index offset (scaled by the array element size) to the base address").

The feature of having a software environment would be obvious for the reasons set forth in the rejection of claim 14.

Per claim 20:

The rejection of claim 14 is incorporated, and further, Tremblay disclose:

- wherein declaring a dimensional dynamic overlay on the data contained within the one-dimensional array includes coding a dimensional dynamic overlay declaration using extended programming language from within the subroutine (col. 24, lines 55-59

"translates program information 609 into translated instructions that include array access instructions on bus 611... each of which corresponds to one of the translated instructions. Each array access instruction references an element within the array").

Per claim 21:

The rejection of claim 14 is incorporated, and further, Tremblay disclose:

- setting an explicit boundary policy for the declared dimensional dynamic overlay (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value").

Per claim 22:

Tremblay disclose:

- A method of processing a data array (col. 24, line 52 "an array access processor") implemented in a computer-readable medium, comprising:
- providing a software program with at least one block of statements (col. 24, lines 55-57 "translated program information... into translated instructions that include array access instructions on bus");
- dynamically declaring a data array within the block of statements (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions");
- setting an array boundary policy for the data array which is defined with or referenced by some of the block of statements (col. 3, lines 28-31 "a maximum array size boundary

value and a minimum array size boundary value"), wherein the array boundary policy dictates run-time actions of the software program that are executed, if during execution of the software program, the data array is accessed outside its boundaries (col. 3, lines 26-30 "an array boundary... typically zero");

- compiling the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set"); and
- executing the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set").

Tremblay does not explicitly disclose accessing the array within the block of statements.

However, admitted prior art discloses in an analogous computer system accessing the array within the block of statements (Applicant's specification, page 3, lines 14-16 "The array elements are often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of accessing the array within the block of

statements as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by Tremblay. The modification would be obvious because of one of ordinary skill in the art would be motivated to access data in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory") the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 23:

The rejection of claim 22 is incorporated, and further, Tremblay does not explicitly disclose declaring an array access handle; declaring an array size; and declaring a data element type.

However, admitted prior art discloses in an analogous computer system declaring an array access handle (Applicant's specification, page 5, lines 12-13 "handle is... pointer into memory that is used to select an object as a whole"); declaring an array size (Applicant's specification, page 3, lines 17-18 "The array name... size of each dimension... declared"); and declaring a data element type (Applicant's specification, page 3, lines 17-18 "The array name... the element data type... declared").

The feature of declaring an array access handle, an array size, a data element type would be obvious for the reasons set forth in the rejection of claim 22.

Per claim 24:

The rejection of claim 23 is incorporated, and further, Tremblay disclose:

- determining the number of data array dimensions for a problem; and selecting the total number of data array dimensions (col. 25, lines 14-19 "Array access processor... generate an array reference entry ARRAYi 615 and minimum and maximum array size values MINi 617A and MAXi 617B, respectively, associated with that array reference entry ARRAYi 615 that are all stored in associative memory element 614").

Per claim 25:

The rejection of claim 24 is incorporated, and further, Tremblay disclose:

- wherein selecting the total number of data array dimensions includes providing a dimension size expression that is constant. The limitation recited in this claim are similar to those recited in claim 24 and rejected under the same rational set forth in connection with the rejection of claim 24 above.

Per claim 26:

The rejection of claim 24 is incorporated, and further, Tremblay disclose:

- wherein selecting the total number of data array dimensions includes providing a dimension size expression that is evaluated at run-time. The limitation recited in this claim are similar to those recited in claim 24 and rejected under the same rational set forth in connection with the rejection of claim 24 above.

Per claim 27:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- obtaining the array boundary policy setting as an array attribute (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value");
- associating attributes of the declared data array with the array access handle (col. 3, line 60 "an array access instruction identifier"); and
- accessing the data array at run-time using the array access handle and array indices (col. 23, lines 55-58 "getfield-putfield accelerator 146 includes an associative memory that has a first section that holds the indices that function as tags, and a second section that holds the field block pointers").

Tremblay does not explicitly disclose performing run-time allocation of memory to obtain a base address attribute; performing run-time calculation of array size attributes from the declared data array.

However, admitted prior art discloses in an analogous computer performing run-time allocation of memory to obtain a base address attribute (Applicant's specification, page 3, lines 23-24 "An array identifier contains a pointer to the base address of the array in memory"); performing run-time calculation of array size attributes from the declared data array (Applicant's specification, page 3, lines 7-8 "These calculations are run-time calculations" and Applicant's

specification, page 3, lines 17-18 "The array name... array dimension... size of each dimensions... declared").

The feature of performing run-time allocation of memory and performing run-time calculation of array size attributes would be obvious for the reasons set forth in the rejection of claim 22.

Per claim 28:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- obtaining the attributes of the declared data array using the array access handle; and performing nm-time boundary policy enforcement based on array access attributes. The limitation recited in this claim are similar to those recited in claim 27 and rejected under the same rational set forth in connection with the rejection of claim 27 above.

Per claim 29:

The rejection of claim 28 is incorporated, and further, Tremblay does not explicitly disclose obtaining the attributes of the declared data array using the array access handle; and performing run-time boundary policy enforcement based on array access attributes.

However, admitted prior art discloses in an analogous computer terminating the program if the array boundary policy aborts for an invalid index (Applicant's specification, page 7, lines 19-21 "If out-of-bounds array indices are used, the underlying array access mechanism may access memory outside that allocated for the array, possibly causing a running program to terminate").

The feature of terminating the program if the array boundary policy aborts for an invalid index would be obvious for the reasons set forth in the rejection of claim 22.

Per claim 30:

The rejection of claim 27 is incorporated, and further, Tremblay disclose:

- applying the array boundary policy to constrain the invalid index values to be valid index values (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value").

Tremblay does not explicitly disclose obtaining the attributes of the declared data array using the array access handle; performing run-time invalid index value detection based on array access attributes for the declared data array; applying the array boundary policy to constrain the invalid index values to be valid index values; calculating an offset into the declared data array from the valid index values and the attributes of the declared data array; and adding the offset to the base address attribute to obtain a memory address for accessing the indexed data array element.

However, admitted prior art discloses in an analogous computer obtaining the attributes of the declared data array using the array access handle (Applicant's specification, page 5, lines 12-13 "A handle is usually a pointer into memory that is used to select an object as a whole"); performing run-time invalid index value detection based on array access attributes for the declared data array (Applicant's specification, page 7, lines 19-21 "If out-of-bounds array indices are used, the underlying array access mechanism may access memory outside that allocated for the array, possibly causing a running program to terminate"); calculating an offset into the declared data array from the valid index values and the attributes of the declared data array

(Applicant's specification, page 5, lines 12-13 "one-dimensional array access can be achieved by adding an index offset (scaled by the array element size) to the base address"); and adding the offset to the base address attribute to obtain a memory address for accessing the indexed data array element (Applicant's specification, page 5, lines 12-13 "one-dimensional array access can be achieved by adding an index offset (scaled by the array element size) to the base address").

The feature of obtaining the attributes of the declared data array using the array access handle, performing run-time invalid index value detection, and adding the offset to the base address would be obvious for the reasons set forth in the rejection of claim 22.

Per claim 31:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- wherein setting an explicit array boundary policy furtherincludes setting a reflect-at-boundary policy that reflect array data at a declared boundary (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value" and col. 23-24, lines 66-67 to 1-2 "Bounds check unit 147 (FIG. 1) in execution unit 140... checks each access to an element of an array to determine whether the access is to a location within the array").

Per claim 32:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary (col. 24, lines 2-4

"When the access is to a location outside the array, bounds check unit 147 issues an active array bound exception signal to execution unit 140").

Per claim 33:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses (col. 24, lines 22-25 "If the value associated with the access of the array's element is less than or equal to the stored maximum value and greater than or equal to the stored minimum value, neither comparator element generates an output signal").

Per claim 34:

Tremblay disclose:

- A method of processing a data array (col. 24, line 52 "an array access processor") implemented in a computer-readable medium, comprising:
- providing a software program with a block of statements in at least one subroutine (col. 24, lines 55-57 "translated program information... into translated instructions that include array access instructions on bus");
- declaring a dimensional dynamic overlay on data contained within a one dimensional array from within the block of statements to initialize attributes within an array attribute storage object (col. 25, lines 11-14 "array access processor 612 receives an array

definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions");

- setting an array boundary policy for the data array which is defined with or referenced by some of the block of statements (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value"), wherein the array boundary policy dictates run-time actions of the software program that are executed, if during execution of the software program, the data array is accessed outside its boundaries (col. 3, lines 26-30 "an array boundary... typically zero");
- compiling the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set"); and
- executing the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set").

Tremblay does not explicitly disclose accessing the array within the block of statements.

However, admitted prior art discloses in an analogous computer system accessing the array within the block of statements (Applicant's specification, page 3, lines 14-16 "The array

elements are often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of accessing the array within the block of statements as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by Tremblay. The modification would be obvious because of one of ordinary skill in the art would be motivated to access data in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose wherein the dimensional dynamic overlay provides a dimensional view on the one-dimensional array, the dimensional dynamic overlap being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose wherein the dimensional dynamic overlay provides a dimensional view on the one-dimensional array (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory"), the dimensional dynamic overlap being capable of providing a view

of at least two dimensions on the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of dimensional dynamic overlay provides a dimensional view on the one-dimensional array, the dimensional dynamic overlap being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 35:

The rejection of claim 34 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a reflect-at-boundary policy that reflect array data at a declared boundary. The limitation recited in this claim are similar to those recited in claim 31 and rejected under the same rational set forth in connection with the rejection of claim 31 above.

Per claim 36:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary. The limitation recited in this claim are similar to those recited in claim 32 and rejected under the same rational set forth in connection with the rejection of claim 32 above.

Per claim 38:

Tremblay disclose:

- A method of processing a data array (col. 24, line 52 "an array access processor") implemented in a computer-readable medium, comprising:
- providing a software program with a block of statements in at least one subroutine (col. 24, lines 55-57 "translated program information... into translated instructions that include array access instructions on bus");
- declaring a dimensional dynamic array from within the block of statements to initialize attributes within an array attribute storage object (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions");
- dynamically allocating memory storage for the declared dimensional dynamic array (col. 25, lines 19-23 "array access processor 612 is simply instructions that are executed by execution unit 616 to load the identifier, and minimum and maximum array size values into associative memory element 614");

- dynamically declaring a data array within the block of statements (col. 25, lines 11-14 "array access processor 612 receives an array definition instruction on bus 611 and an array access instruction identifier on bus 613 of the translated software instructions");
- setting an array boundary policy for the data array which is defined with or referenced by some of the block of statements (col. 3, lines 28-31 "a maximum array size boundary value and a minimum array size boundary value"), wherein the array boundary policy dictates run-time actions of the software program that are executed, if during execution of the software program, the data array is accessed outside its boundaries (col. 3, lines 26-30 "an array boundary... typically zero");
- compiling the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set"); and
- executing the software program (col. 6, lines 45-50 "A JAVA compiler JAVAC, (FIG. 2) that is executing on a computer platform, converts an application 201 written in the JAVA computer language to an architecture neutral object file format encoding a compiled instruction sequence 203, according to the JAVA Virtual Machine Specification, that includes a compiled instruction set").

Tremblay does not explicitly disclose providing a dynamic overlay storage object with attributes assigned from the declared dimension dynamic array; accessing data from the dynamically

allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object; automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared; and automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared.

However, admitted prior art discloses in an analogous computer system providing a dynamic overlay storage object with attributes assigned from the declared dimension dynamic array (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared storage that is only available after its declaration, and only within the scope of a block of statements in which it is declared"); accessing data from the dynamically allocated memory storage as a dimensional indexed array from within the block of statements using the dynamic overlay storage object (Applicant's specification, page 3, lines 14-16 "The array elements are often stored contiguously in the computer's memory, and the subscript or index of the first element is normally zero in all the dimensions applied to the array"); automatically freeing the dynamically allocated memory storage when leaving a subroutine in which the dynamic array is declared (Applicant's specification, page 4, lines 26-28 "Automatic storage is declared... storage is released when code execution leaves the block"); and automatically freeing the dynamic overlay storage object when leaving a subroutine in which the dynamic array is declared (Applicant's specification, page 4, lines 30-31 "Subroutines are used in C to dynamically allocate a contiguous region of memory and free it after it is no longer needed").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of providing a dynamic overlay storage object

accessing data from the dynamically allocated memory, automatically freeing the dynamically allocated memory as taught in admitted prior art in corresponding to the method of accessing a memory array as taught by Tremblay. The modification would be obvious because of one of ordinary skill in the art would be motivated to providing a dynamic overlay storage object accessing data from the dynamically allocated memory, automatically freeing the dynamically allocated memory in an array to simply and explicitly specify array access behavior for invalid array access indices as suggested in admitted prior art (Applicant's specification, page 8-9, lines 32-34 and 1-2).

Neither Tremblay nor Admitted Prior Art explicitly disclose including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array.

However, Spencer discloses in an analogous computer system disclose including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations (col. 9, lines 8-23 "resize( ) function... implemented... subclass, so that a call to the two-dimensional resize( ) function... will in turn resize each of the one-dimensional arrays of which it (the two-dimensional array) is composed... data is required, this could be easily done through another function, such as the grow( ) function, which could ensure that the new size is greater than or equal to the current size in every dimension, and then could copy the data before freeing the previously used memory") the dimensional dynamic overlay being capable of providing a view of at least two dimensions on

the one-dimensional array (col. 10, lines 35-40 "sparse array in that a multidimensional array is transformed into a one-dimensional array...").

Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of including a dimensional dynamic overlay to provide a dimensional view on a one-dimensional array associated with contiguous memory locations the dimensional dynamic overlay being capable of providing a view of at least two dimensions on the one-dimensional array as taught by Spencer into the method of accessing memory array as taught by the combination system of Tremblay and Admitted Prior Art. The modification would be obvious because of one of ordinary skill in the art would be motivated to define dynamic overlay within the subroutine to direct control of memory allocation access to increase the efficiency, accuracy, and reliability of the programs as suggested by Spencer (col. 4, lines 35-43).

Per claim 37:

The rejection of claim 34 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses. The limitation recited in this claim are similar to those recited in claim 33 and rejected under the same rational set forth in connection with the rejection of claim 33 above.

Per claim 39:

The rejection of claim 38 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a reflect-at-boundary policy that reflect array data at a declared boundary. The limitation recited in this claim are similar to those recited in claim 31 and rejected under the same rational set forth in connection with the rejection of claim 31 above.

Per claim 40:

The rejection of claim 22 is incorporated, and further, Tremblay disclose:

wherein setting the array boundary policy further includes setting a confined index and boundary policy that replicates data beyond a detected boundary. The limitation recited in this claim are similar to those recited in claim 32 and rejected under the same rational set forth in connection with the rejection of claim 32 above.

Per claim 41:

The rejection of claim 38 is incorporated, and further, Tremblay disclose:

- wherein setting the array boundary policy further includes setting a pre-defined array attribute value that is to be read for all out of bounds accesses. The limitation recited in this claim are similar to those recited in claim 33 and rejected under the same rational set forth in connection with the rejection of claim 33 above.

Per claim 43:

The rejection of claim 42 is incorporated, and further, Tremblay disclose:

- wherein the translator includes a compiler adapted for converting the extended language into machine code instructions that is able to be run on the processor (col. 5,lines 43-46 "emulating the JAVA virtual machine as a software interpreter, compiling JAVA virtual machine instructions (either in batch or just-in-time) to machine instruction native to a particular hardware processor").

Per claim 44:

The rejection of claim 42 is incorporated, and further, Tremblay disclose:

- a language converter for converting the extended language into a mid-level programming code; and a compiler for compiling the mid-level language program code into machine code instructions that is able to be rtm on the processor. The limitation recited in this claim are similar to those recited in claim 43 and rejected under the same rational set forth in connection with the rejection of claim 43 above.

Per claim 45:

The rejection of claim 42 is incorporated, and further, Tremblay disclose:

- the language converter includes a converter program encoded on the computer readable medium; and the processor executes the converter program to read the extended language and convert the extended language into the mid-level programming code. The limitation recited in this claim are similar to those recited in claim 43 and rejected under the same rational set forth in connection with the rejection of claim 43 above.

Per claim 46:

The rejection of claim 42 is incorporated, and further, Tremblay does not explicitly disclose the extended language is a C programming language with language extensions; and the language converter converts the extended language into C code.

However, admitted prior art discloses in an analogous computer the extended language is a C programming language with language extensions (Applicant's specification, page 1, line 30 "Examples of mid-level programming languages are C and C++"); and the language converter converts the extended language into C code (Applicant's specification, page 2, lines 12-14 "Files can be translated into low-level machine code or assembler for a target computer type by a computer program called a compiler").

The feature of using C language and a language converter would be obvious for the reasons set forth in the rejection of claim 42.

Per claim 50:

The rejection of claim 48 is incorporated, and further, Tremblay does not explicitly disclose wherein the translator includes a cross compiler.

However, admitted prior art discloses in an analogous computer wherein the translator includes a cross compiler (Applicant's specification, page 2, lines 14-15 "A cross compiler is a compiler that runs on one computer and produces machine code targeted for a different type of computer").

The feature of using cross compiler would be obvious for the reasons set forth in the rejection of claim 48.

Per claim 51:

The rejection of claim 48 is incorporated, and further, Tremblay does not explicitly disclose wherein the translator includes a native compiler.

However, admitted prior art discloses in an analogous computer wherein the translator includes a native compiler (Applicant's specification, page 2, lines 15-16 "A native compiler runs on the target computer or a computer of the same type").

The feature of using native compiler would be obvious for the reasons set forth in the rejection of claim 48.

Claim 42, 47, 48, and 49 are the system claims corresponding to method claim 22 and rejected under the same rational set forth in connection with the rejection of claim 22 above.

Response to Arguments

9. Applicant's arguments with respect to claims have been considered but they are not persuasive.

In the remarks, the applicant has argued that:

(i) Applicants do not believe the cited background statements read on the claim language that states "accessing data from within the block of statements as a dimensional indexed array using the array attribute storage object." This does not lead to the conclusion that accessing an array via an index value is that

same as the language in claim 1, where the access data is achieved via an "attribute storage object."

(ii) Applicants unable to find in the Trembley reference a dimensional dynamic overlay to provide a dimensional view (capable of at least a two dimensional view) into one-dimensional arrays as Applicants' independent claims now positively recited.

Examiner's response:

(i) In response to Applicant's arguments regarding the background language is the same as the limitation "accessing data... attribute storage object" as recited in claim 1. As noted by the Applicants that the background section simply states arrays are often accessed via an index value and supported by an example that an array A having 2 elements may be referenced with the statement A[0] to acquire the first element of that array A, this is a common practice among the programmers (Remarks, page 14). Examiner do not believe this is any different than the limitation recited in claim 1, because accessing the stored data using indexing technique which is very common among the programmers as indicated by the Applicants (Remarks, page 14). Applicants only make general allegations. Therefore, the rejection is proper and maintained herein.

(ii) In response to Applicant's arguments regarding "Trembley reference a dimensional dynamic overlay to provide a dimensional view (capable of at least a two dimensional view) into one-dimensional arrays as Applicants'

independent claims now positively recited" has been considered but are moot in view of new ground(s) of rejection.

Conclusion

10. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to **Satish S. Rampuria** whose telephone number is **(571) 272-3732**. The examiner can normally be reached on **8:30 am to 5:00 pm** Monday to Friday except every other Friday and federal holidays. Any inquiry of a general nature or relating to the status of this application should be directed to the **TC 2100 Group receptionist: 571-272-2100**

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, **Tuan Q. Dam** can be reached on **(571) 272-3695**. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Satish S. Rampuria
Patent Examiner
Art Unit 2191
07/11/2005

WEI Y. ZHEN
PRIMARY EXAMINER